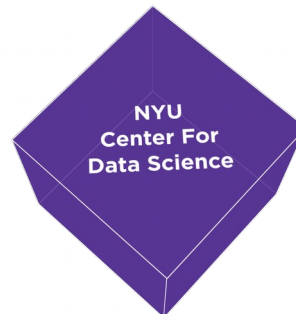


What's new in

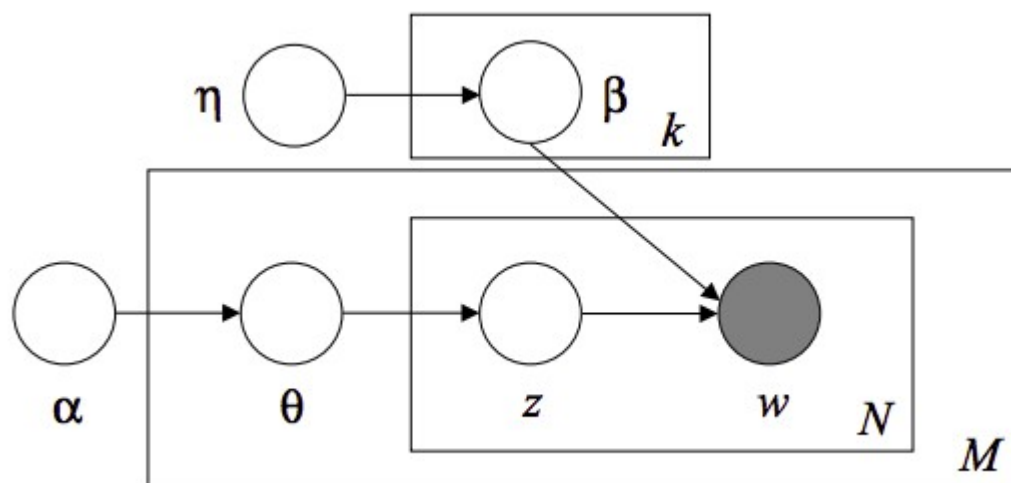


Andreas Mueller



Latent Dirichlet Allocation

using online variational inference



Topic #0:

government people mr law gun state president states public use right
rights national new control american security encryption health
united

Topic #1:

drive card disk bit scsi use mac memory thanks pc does video hard
speed apple problem used data monitor software

Topic #2:

said people armenian armenians turkish did saw went came women
killed children turkey told dead didn left started greek war

Topic #3:

year good just time game car team years like think don got new play
games ago did season better ll

Topic #4:

10 00 15 25 12 11 20 14 17 16 db 13 18 24 30 19 27 50 21 40

Topic #5:

windows window program version file dos use files available display
server using application set edu motif package code ms software

Topic #6:

edu file space com information mail data send available program ftp
email entry info list output nasa address anonymous internet

Topic #7:

ax max b8f g9v a86 pl 145 1d9 0t 34u 1t 3t giz bhj wm 2di 75u 2tm
bxn 7ey

Topic #8:

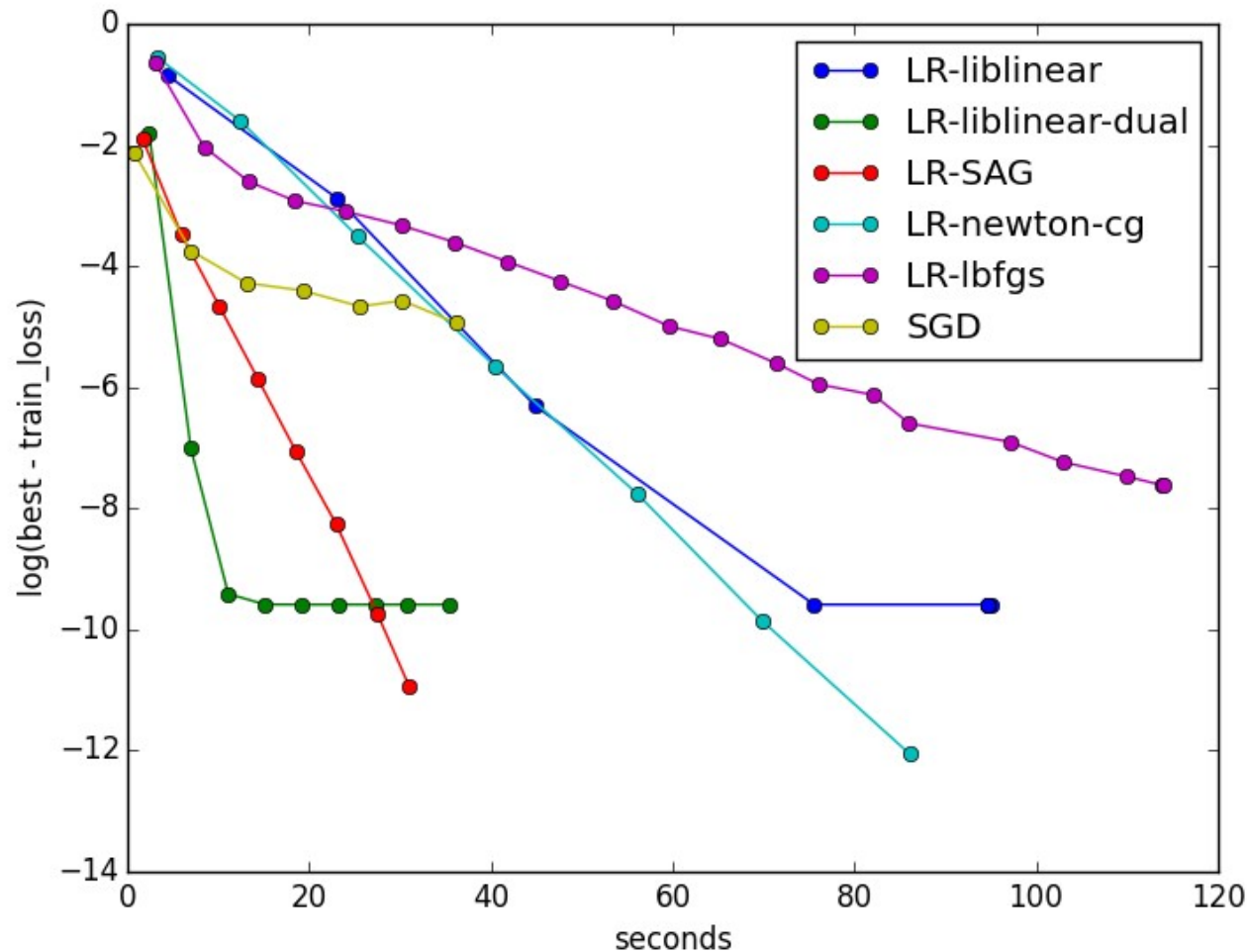
god people jesus believe does say think israel christian true life jews
did bible don just know world way church

Topic #9:

don know like just think ve want does use good people key time way
make problem really work say need

By Chyi-Kwei Yau, based on code by Matt Hoffman

SAG for Logistic Regression and Ridge Regression



By Danny Sullivan and Tom Dupre la Tour

Coordinate Descent Solver for Non-Negative Matrix Factorization

Topics in NMF model:

Topic #0:

don people just like think know time good right ve make say want did really way new use going said

Topic #1:

windows file dos files window program use running using version ms problem server pc screen ftp run application os software

Topic #2:

god jesus bible christ faith believe christians christian heaven sin hell life church truth lord say belief does existence man

Topic #3:

geb dsl n3jxp chastity cadre shameful pitt intellect skepticism surrender gordon banks soon edu lyme blood weight patients medical probably

Topic #4:

key chip encryption clipper keys escrow government algorithm secure security encrypted public des nsa enforcement bit privacy law secret use

Topic #5:

drive scsi ide drives disk hard controller floppy hd cd mac boot rom cable internal tape bus seagate bios quantum

Topic #6:

game team games players year hockey season play win league teams nhl baseball player detroit toronto runs pitching best playoffs

Topic #7:

thanks mail does know advance hi info looking anybody address appreciated help email information send ftp post interested list appreciate

Topic #8:

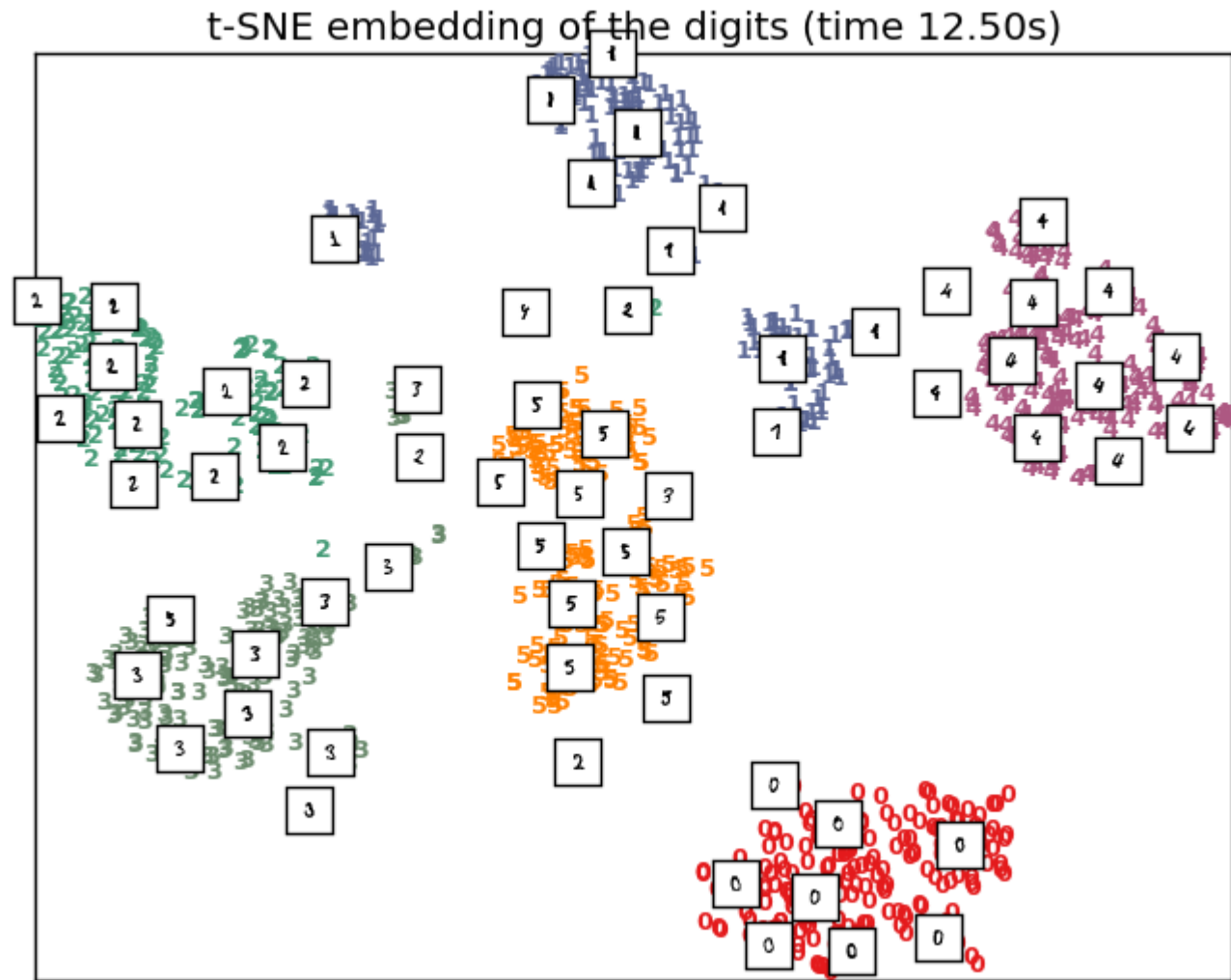
card video monitor vga bus drivers cards color driver ram ati mode memory isa graphics vesa pc vlb diamond bit

Topic #9:

00 sale 50 shipping 20 10 price 15 new 25 30 dos offer condition 40 cover asking 75 interested 01

By Tom Dupre la Tour and Mathieu Blondel

Barnes-Hut Approximation for T-SNE manifold learning



FunctionTransformer

```
>>> import numpy as np
>>> from sklearn.preprocessing import FunctionTransformer
>>> transformer = FunctionTransformer(np.log1p)
>>> X = np.array([[0, 1], [2, 3]])
>>> transformer.transform(X)
array([[ 0.          ,  0.69314718],
       [ 1.09861229,  1.38629436]])
```

VotingClassifier

```
>>> clf1 = LogisticRegression(random_state=1)
>>> clf2 = RandomForestClassifier(random_state=1)
>>> clf3 = GaussianNB()

>>> eclf = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('gnb', clf3)], voting='hard')
```

Scalers

- RobustScaler
- MaxAbsScaler

By Thomas Unterthiner.

Add Backlinks to Docs



[Home](#) [Installation](#) [Documentation](#) [Examples](#)

Google™ Custom Search

Search x

Previous
sklearn.ense
m...

Up
API
Reference

This documentation is for
scikit-learn **version**
0.18.dev0 — [Other](#)
[versions](#)

If you use the software,
please consider [citing](#)
[scikit-learn](#).

3.2.4.3.1.
`sklearn.ensemble.RandomForestC`
`lassifier`
3.2.4.3.1.1. Examples using
`sklearn.ensemble.RandomForestClas`
`sifier`

3.2.4.3.1. `sklearn.ensemble.RandomForestClassifier`

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None,  
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,  
bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None)  
\[source\]
```

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if `bootstrap=True` (default).

Read more in the [User Guide](#).

Parameters: `n_estimators` : integer, optional (default=10)

The number of trees in the forest.

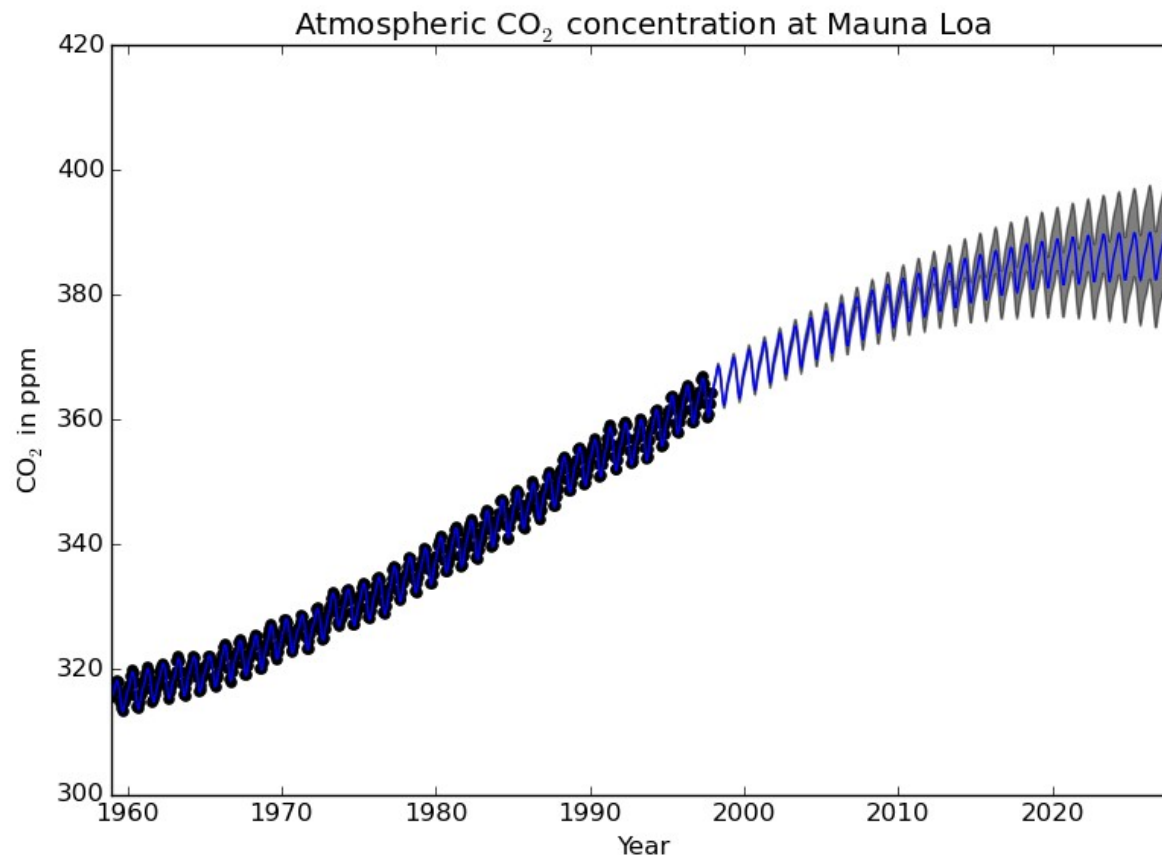
criterion : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

max_features : int, float, string or None, optional (default="auto")

What the future will bring (0.18)

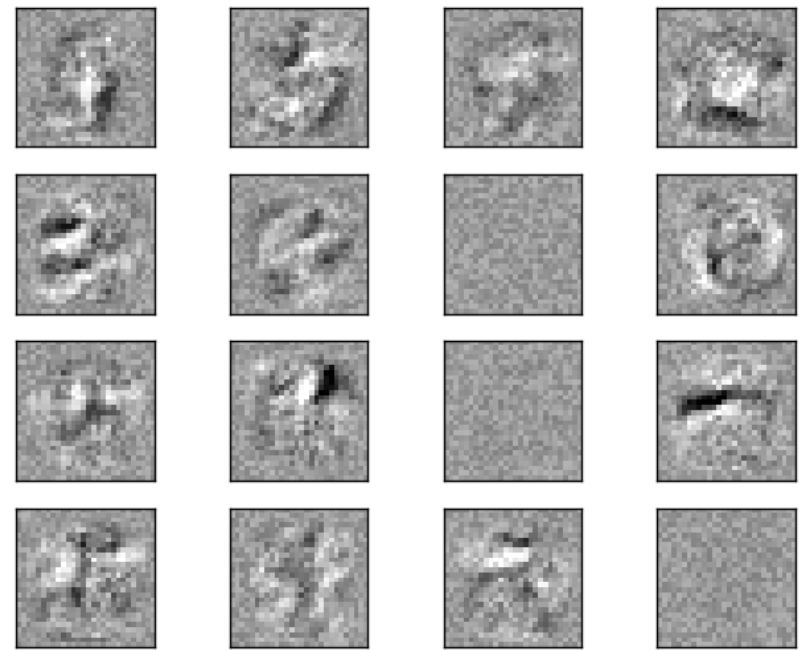
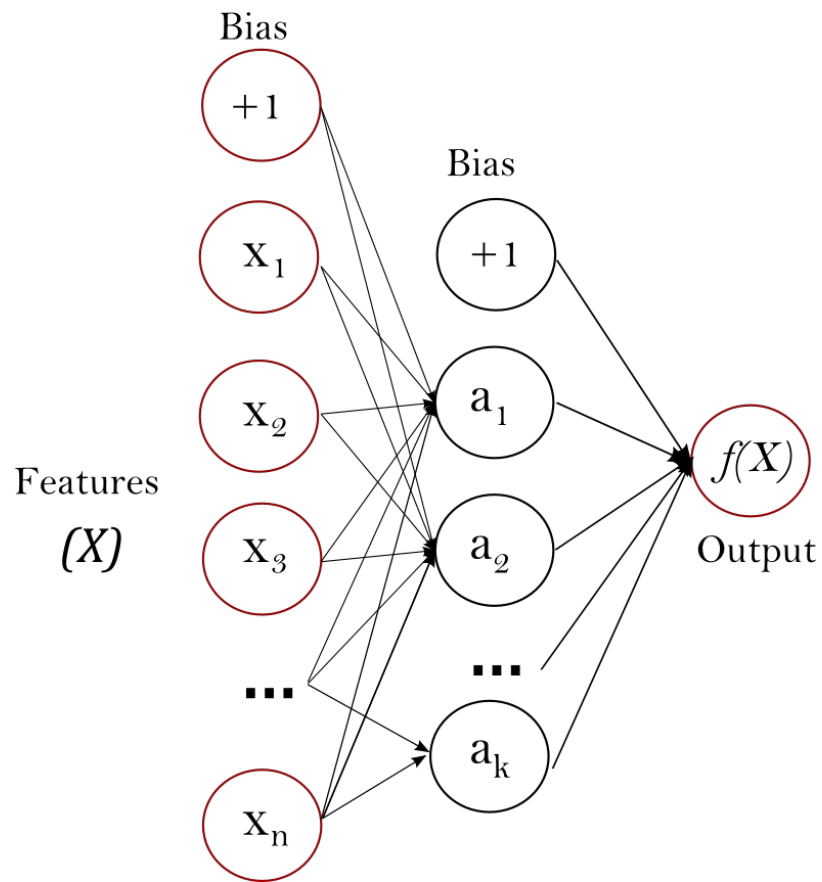
Gaussian Process Rewrite



$34.4^{**2} * \text{RBF}(\text{length_scale}=41.8)$
 $+ 3.27^{**2} * \text{RBF}(\text{length_scale}=180)$
 $* \text{ExpSineSquared}(\text{length_scale}=1.44, \text{periodicity}=1)$
 $+ 0.446^{**2} * \text{RationalQuadratic}(\alpha=17.7, \text{length_scale}=0.957)$
 $+ 0.197^{**2} * \text{RBF}(\text{length_scale}=0.138) + \text{WhiteKernel}(\text{noise_level}=0.0336)$

By Jan Hendrik Metzen.

Neural Networks



By Jiyuan Qian and Issam Laradji

Improved Cross-Validation

current

```
>>> import numpy as np
>>> from sklearn.model_selection import KFold

>>> X = ["a", "b", "c", "d"]
>>> kf = KFold(n_folds=2)
>>> for train, test in kf.split(X):
...     print("%s %s" % (train, test))
[2 3] [0 1]
[0 1] [2 3]
```

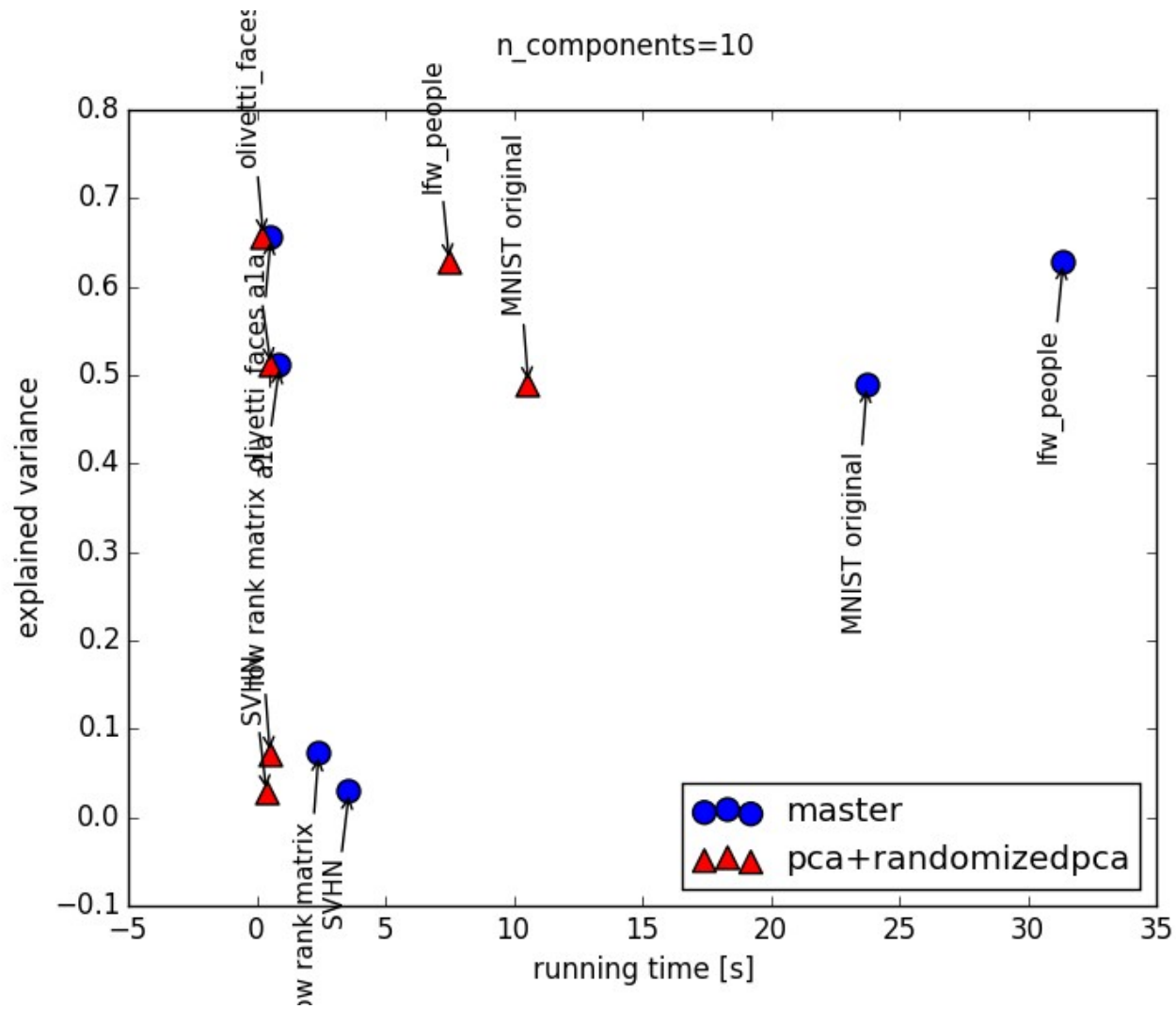
future

```
>>> import numpy as np
>>> from sklearn.cross_validation import KFold

>>> kf = KFold(4, n_folds=2)
>>> for train, test in kf:
...     print("%s %s" % (train, test))
[2 3] [0 1]
[0 1] [2 3]
```

```
gs = GridSearchCV(LinearSVC(random_state=0), param_grid={'C': [1, 10]},
                  cv=inner_cv)
cross_val_score(gs, X=X, y=y, labels=labels, cv=outer_cv,
                fit_params={'labels': labels})
```

Faster PCA



By Giorgio Patrini